Presses
universitaires
de Caen
Maison de la Recherche en
*Sciences Humaines (*mrsh*)*
14032 Caen cedex

Davy
DERMILLY
Année 05/06

ONIX-MANAGER

Manuel d'installation d'onix-manager

Table des matières

1 Onix-manager **1**
- 1.1 Configuration requise . . . 1
- 1.2 Installation . . . 1
  - 1.2.1 Variables d'environnement . . . 1
  - 1.2.2 Installer eXist . . . 1
  - 1.2.3 Installer onix-manager . . . 2
- 1.3 Configuration . . . 2
  - 1.3.1 Le fichier conf.properties . . . 2
  - 1.3.2 Le fichier log4j.properties . . . 4
  - 1.3.3 Le fichier ApplicationRessources.properties . . . 4
  - 1.3.4 Les fichiers de configuration XML . . . 4

2 Service web dédié à spip coté serveur **9**
- 2.1 Configuration requise . . . 9
- 2.2 Installation . . . 9
- 2.3 Configuration du service web . . . 10

3 Service web dédié à spip coté client **11**
- 3.1 Configuration requise . . . 11
- 3.2 Configuration du service web . . . 11

4 Service web répondant au moteur de recherche **12**
- 4.1 Configuration requise . . . 12
- 4.2 Installation . . . 12
  - 4.2.1 Cas où axis n'est pas installé . . . 12
  - 4.2.2 Cas où axis est déjà installé . . . 12
- 4.3 Configuration du service web . . . 13
  - 4.3.1 Le fichier onixcatalogws.properties . . . 13
  - 4.3.2 Le fichier droits.xml . . . 13
  - 4.3.3 Pour finir . . . 14

Chapitre 1

Onix-manager

1.1 Configuration requise

- **jre-1_5_0_10-linux-i586.bin**(http ://java.sun.com/javase/downloads/index_jdk5.jsp)
- **exist-1.0rc.war**(http ://exist.sourceforge.net/index.html#download)
- **apache-tomcat-5.5.17**(http ://tomcat.apache.org/download-55.cgi#5.5.20)
- **onix-manager.war**

1.2 Installation

1.2.1 Variables d'environnement

Dans le fichier **.bashrc** de l'utilisateur qui lance **Tomcat**, déclarer les variables suivantes :

- export JAVA_HOME="répertoire ou est installé Java" (ex :/opt/apache-tomcat-5.5.17)
- export CATALINA_HOME="répertoire ou est installé tomcat" (ex : /opt/-jdk)

1.2.2 Installer eXist

Pour l'installation d'eXist il faut placer l'archive **exist-1.0rc.war** dans le répertoire webApps de **Tomcat**. **Tomcat** la décompacte automatiquement. Une fois l'archive decompactée, renommer le répertoire *exist-1.0rc* en *exist* puis se rendre sur la page d'accueil d'eXist qui pourrait être :

http://localhost:8080/exist/index.xml

Dans le menu *Administration/admin*, entrer un nouvel utilisateur et mot de passe (*Manage users*) ou alors "admin" et mot de passe. On utilisera ce *login* dans la configuration d'**onix-manager**.

Dans le menu *Browse Collections*, créer une nouvelle collection qui contiendra le catalogue principal exemple : *bassinOnix*

Entrer dans cette collection et ajouter autant de catalogues et catégories que souhaité. Les garnir en y incluant des fichiers XML au format ONIX (histoire, géographie, politique...).

Attention, il y a un problème avec la gestion de la DTD distante dans eXist.

Il faut soit :
– supprimer le DOCTYPE du fichier à intégrer,
– ou bien, récupérer la DTD ONIX[1] et la mettre en local.
Pour le nommage des collections, il est préférable de procéder comme suit :
– **Nom_de_la_collection**
Les espaces sont remplacés par des tirets et les caractères ne sont pas accentués.

1.2.3 Installer onix-manager

Pour installer **onix-manager**, il suffit de mettre **onix-manager.war** dans le répertoire *webApps* de **Tomcat** qui se chargera de déployer l'application automatiquement.

1.3 Configuration

1.3.1 Le fichier conf.properties

Le fichier *conf.properties* se trouve dans le répertoire **WEB-INF/classes** d'**onix-manager**.
paramètres pour eXist :

```
#Url du serveur qui héberge exist
serveur.url=//localhost:8080

#Identifiant de connexion à la base exist (nom de l'utilisateur créé
    pour qu'onix-manager se connecte à eXist)
application.login=admin
application.password=mdp

# Nom de la collection principale dans eXist où est rangé le catalogue
collection=bassinOnix

#Nombre maximal d'enregistrements renvoyé par chaque recherche
searchResult.numberOfRecord=15

# Numéro du dernier enregistrement ajouté à la base (doit se terminer
    par un chiffre pour l'auto-incrémentation)
product.lastRecordReference=p1
```

[1] sur le site de editeur.org

Fichiers de configuration de l'application

Onix-manager fournit les fichiers de configurations suivants :

- *droits.xml*
- *productForPass.xml*
- *defaultProduct.xml*
- *codeList.xml*

Ils sont présents dans le répertoire WEB-INF d'**onix-manager**.

Il va falloir renseigner leur emplacement dans le fichier *conf.properties* :

```
# Emplacement du fichier des droits et des profils utilisateurs

userFile.xml.uri=votre tomcat/webapps/onix-manager/WEB-INF/
    droits.xml

# Emplacement du fichier qui contient le produit de référence avec tous
    les champs utilisés par l'application
productForPass.xml.uri=votre
tomcat/webapps/onix-manager/WEB-INF/productForPass.xml

# Emplacement du fichier d'un produit par défaut (il s'agit d'un produit
    préalablement renseigné qui apparaîtra tel quel lors de l'ajout d'
    un nouveau produit).

defaultProduct.xml.uri=votre tomcat/webapps/onix-manager/WEB-
    INF/defaultProduct.xml

# Emplacement du fichier des listes de codes ONIX
codeList.xml.uri=votre tomcat/webapps/onix-manager/WEB-INF/
    codeList.xml
```

Pour l'upload des images

```
# Le chemin du répertoire ou sont stockées les images des différents
    produits
pictures.directory.location=votre repertoire

# Url à laquelle ces images seront visualisables avec un navigateur
    Internet
pictures.directory.url=http://votre repertoire
```

Il faut bien sûre que ce répertoire ait été préalablement créé.

> Attention aux droits en écriture sur ce répertoire si on est sous **Linux**.

1.3.2 Le fichier log4j.properties

Le fichier **log4j.properties** se trouve dans le répertoire *WEB-INF/Classes* d'**onix-manager**.

Dans ce fichier, il faut juste renseigner la ligne suivante :

```
# second: Any configuration information needed for that appender.
log4j.appender.rolling.File=../logs/onix-manager.log
```

Mettre à cet endroit le chemin du fichier log de l'application (créer un fichier .log vide s'il n'existe pas).

C'est dans ce fichier que figureront les erreurs si l'application ne fonctionne pas.

1.3.3 Le fichier ApplicationRessources.properties

Dans ce fichier sont stockés les différents titres et labels de l'application. Par exemple, si l'on veut une interface en anglais, il faut éditer ce fichier et renseigner tous les champs en anglais.

1.3.4 Les fichiers de configuration xml

droits.xml C'est ici que sont définis les droits des utilisateurs. Il existe quatre sortes de droits :

- "R" autorisé en lecture
- "W" autorisé en écriture
- "NO-R" interdit en lecture
- "NO-W" interdit en écriture

Au sein de ce fichier, on peut créer des *"profil"* et des *"user"*.

Un exemple de *profil* :

```
<profil name="relecteur">
  <pass type="R">
    <path>//Product/*</path>
    <path>//Product/descendant::*</path>
  </pass>
  <pass type="W">
    <path>//Product/Title/TitleText</path>
    <path>//Product/OtherText/*</path>
  </pass>
</profil>
```

Un exemple de *user* :

```
<user login="dupont" password="123" lastname="DUPONT" firstname
    ="George"
  function="relecteur" websiteManagement="false">
  <profilName>relecteur</profilName>
  <pass type="W">
    <path>//Product/Subject/*</path>
  </pass>
  <pass type="NO-W">
    <path>//Product/Subject/SubjectSchemeIdentifier</path>
  </pass>
  <pass type="NO-R">
    <path>//Product/Contributor</path>
  </pass>
</user>
```

Explications Ici, George DUPONT est un *relecteur* qui ne peut pas mettre à jour le site Internet distant (*websiteManagement="false"* via service web SPIP etc.).

Il récupère tous les droits du profil "relecteur" (*<profilName>relecteur</profilName>*). C'est-à-dire qu'il peut lire tout ce qu'il y dans un *Product* et ses enfants et qu'il ne peut modifier que les textes de titre (*//Product/Title/TitleText*) et les résumés (*//Product/OtherText/**).

On peut tout de même lui ajouter des droits en dehors de son profil. Ce qui veut dire que dans l'exemple ci-dessus, George DUPONT peut également modifier les sujets (*//Product/Subject/** en "W") sauf les types de sujet (*//Product/Subject/SubjectSchemeIdentifier* en "NO-W") et qu'il ne peut pas voir les auteurs (*//Product/Contributor* en "NO-R").

On peut ainsi créer divers profils et un utilisateur peut en hériter de plusieurs. En effet, monsieur DUPONT peut avoir de nouvelles fonctions comme par exemple la gestion de stock. Ce qui donnerait :

```
  <profil name="relecteur">
    <pass type="R">
      <path>//Product/*</path>
      <path>//Product/descendant::*</path>
    </pass>
    <pass type="W">
      <path>//Product/Title/TitleText</path>
      <path>//Product/OtherText/*</path>
    </pass>
  </profil>

<-- Ajouter un profil "gestion stock"

  <profil name="gestion stock">
    <pass type="R">
      <path>//Product/SupplyDetail/Stock/*</path>
    </pass>
    <pass type="W">
      <path>//Product/SupplyDetail/Stock/*</path>
    </pass>
  </profil> -->

  <user login="dupont" password="123" lastname="DUPONT" firstname
      ="George"
    function="relecteur" websiteManagement="false">
    <profilName>relecteur</profilName>

   <-- Ajouter la reference au profil "gestion stock"
    <profilName>gestion stock</profilName> -->

    <pass type="W">
      <path>//Product/Subject/*</path>
    </pass>
    <pass type="NO-W">
      <path>//Product/Subject/SubjectSchemeIdentifier</path>
    </pass>
    <pass type="NO-R">
      <path>//Product/Contributor</path>
    </pass>
  </user>
```

productForPass.xml Ce fichier contient tous les champs gérés par l'application. Le modifier, c'est prendre le risque que l'application ne fonctionne plus correctement. Il est donc conseillé d'en garder une copie avant toute modification.

On peut remarquer que les champs sont pré-remplis :

```
<Product>
  <RecordReference>111</RecordReference>
  <NotificationType>03</NotificationType>
  <ProductIdentifier>
    <ProductIDType>02</ProductIDType>
    <IDValue>Valeur premier ID</IDValue>
  </ProductIdentifier>
  ...
</Product>
```

Il est possible et même conseillé de changer la valeur de ces champs.

Car lorsque l'on voudra ajouter, par exemple, un nouvel identifiant au produit que nous sommes en train de modifier via **onix-manager**, ce champ sera pré-rempli en fonction de ce qu'**onix-manager** trouvera dans le fichier *productForPass.xml*. Dans notre exemple, on aura "Valeur premier ID".

Attention à ce que les codes ("02","03""AB",etc.) soient en accords avec la liste des codes de la DTD ONIX

.

defaultProduct.xml Ce fichier est basé sur le même principe que le précédent le caractère sensible en moins.

Il contient juste un produit par défaut avec certains champs pré-renseignés. Ainsi, lorsqu'on ajoute un nouveau produit avec **onix-manager**, celui-ci est déjà renseigné avec des valeurs par défaut (ce sont en générales des valeurs communes à chaque produit).

codeList.xml Ce fichier nous est fournit par ONIX et contient toutes les listes de codes ONIX nécessaires aux listes déroulantes que propose **onix-manager** lors de la modification d'un produit.

On peut, par exemple, retirer d'une liste de type d'identifiants de produit, les codes d'identifiants que l'on utilise pas :

```
</CodeList>
   <CodeListNumber>5</CodeListNumber>
   <CodeListDescription>Code type identifiant produit</
       CodeListDescription>
   <IssueNumber>0</IssueNumber>

  <Code>
    <CodeNumber>3</CodeNumber>
    <CodeValue>03</CodeValue>
    <CodeDescription>EAN.UCC-13</CodeDescription>
    <CodeNotes>EAN-UCC Article Number (13 positions)</
        CodeNotes>
    <IssueNumber>0</IssueNumber>
  </Code>
   ...
<-- début suppression
  <Code>
    <CodeNumber>5</CodeNumber>
    <CodeValue>05</CodeValue>
    <CodeDescription>ISMN</CodeDescription>
    <CodeNotes>International Standard Music Number...</CodeNotes>
    <IssueNumber>0</IssueNumber>
  </Code>
   fin suppression -->
   ...
  <Code>
    <CodeNumber>9</CodeNumber>
    <CodeValue>15</CodeValue>
    <CodeDescription>ISBN-13</CodeDescription>
    <CodeNotes>International Standard Book Number...</CodeNotes>
    <IssueNumber>4</IssueNumber>
  </Code>
</CodeList>
```

Il est possible de modifier le champ *CodeDescription* qui figurera dans les listes déroulantes. Attention à ne pas modifier le champ *CodeValue* pour rester dans la norme ONIX.

Chapitre 2

Service web dédié à spip coté serveur

2.1 Configuration requise

- PHP5 installé.
- MySQL4 installé.
- SPIP installé.
- **OnixSpipWS.gz**

2.2 Installation

Décompacter l'archive **OnixSpipWS.gz** et placer les fichiers dans le dossier **www** du serveur web **Apache**.

2.3 Configuration du service web

Ouvrir le fichier **OnixSpipWSConstants.php** et renseigner les différentes constantes.

```
define(SERVEUR,'localhost'); nom du serveur MySQL
define(DB_USER,"root");      login pour la base SPIP
define(DB_PASSWORD,"mldacm"); mdp correspondant au login
define(DB,"spip");           Nom de la base spip
define(SPIP_TABLES_PREFIX,"spip"); (spip par défaut)

/*
 Lorsque l'on saisi un nouvel article dans SPIP, on peut
le consulter via une URL qui se termine par le numéro de l'article.
 Il faut mettre ci-dessous cette URL sans le numéro.
*/
define(URL_ARTICLE,"http://localhost/spip/spip.php?article");

// Numéro de rubrique dans laquelle figure le catalogue
define(CATALOGUE_ID_RUBRIQUE,"2");

// entrer ici le login et mdp de l'utilisateur de ce service web
define(LOGIN,"davy");
define(PASSWORD,"123");
```

Pour les tests, lancer *index.php* qui interroge le service Web.

Chapitre 3

Service web dédié à spip coté client

3.1 Configuration requise

– **onix-manager.war** installé.

3.2 Configuration du service web

Dans le fichier **conf.properties** d'**onix-manager** il faut renseigner les lignes suivantes :

```
#
#   WEB SERVICE MISE A JOUR DU SITE INTERNET
#

# Si l'application  utilise  le  web service  dédié  à  SPIP ou non

onixSpipWSServer.used=false # mettre à true


# URL du fichier OnixSpipWSServer.php (dans le dossier OnixSpipWS
    dans
le  dossier  www de votre serveur Apache)
onixSpipWSServer.url=http://localhost/webService/OnixSpipWSServer.
    php

# Login et mot de passe de l'application  pour qu'elle  puisse  utiliser
le  web service  dédié  à  SPIP (correspond à ceux du fichier
    OnixSpipWSConstants.php)
onixSpipWSServer.login=davy
onixSpipWSServer.password=123
```

Chapitre 4

Service web répondant au moteur de recherche

4.1 Configuration requise

- **jre-1_5_0_10-linux-i586.bin** installé
- **exist-1.0rc.war** installé
- **apache-tomcat-5.5.17** installé
- **axis.gz**

4.2 Installation

l'installation de ce service web passe par diffétentes étapes :

4.2.1 Cas où axis n'est pas installé

1. décompacter l'archive *axis.gz* et placer le dossier obtenu dans le répertoire *webApps* de **Tomcat**.
2. modifier le fichier *onixcatalogws.properties*
3. modifier le fichier *droits.xml*

4.2.2 Cas où axis est déjà installé

Après avoir décompacté l'archive *axis.gz* :

1. Récupérer le dossier *axis décompacté/WEB-INF/classes/onixcatalogws* et le fichier *axis décompacté/WEB-INF/classes/onixcatalogws.properties* et les mettre dans le dossier */WEB-INF/classes* de l'axis existant.
2. Récupérer les fichiers :
   - exist.jar
   - onix-exist-tools.jar
   - onix-kernel.jar
   - onix-ws-search-engine.jar
   - xml-db.jar
   - xml-user-manager.jar

- xml-utils.jar
- xmldb.jar
- xmlrpc-1.2-patched.jar

qui se trouvent dans le dossier *axis décompacté/WEB-INF/lib* et les mettres dans le dossier */WEB-INF/lib* de l'axis existant.

3. modifier le fichier *onixcatalogws.properties*
4. modifier le fichier *droits.xml*

4.3 Configuration du service web

4.3.1 Le fichier onixcatalogws.properties

Le fichier *onixcatalogws.properties* se trouve dans le répertoire *WEB-INF/classes* d'**axis**.

```
# Url tomcat
serveur.url=//localhost:8080

# Nom de la collection principale qui contient votre catalogue
exist.onixCollection=bassinOnix

# Login et mot de passe dans eXist pour acceder à la collection "
    bassinOnix"
application.login=admin
application.password=mdp

# Nombre maximal d'enregistrements renvoyés par les recherches afin de
    ne pas saturer le serveur
searchResult.maxNumberOfRecord=20

#Emplacement du fichier xml (droits.xml) contenant le profil "public"
user. file =/path/du/fichier/droits.xml

#login et mot de l' utilisateur qui possède un profil "public" dans le
 fichier "droits.xml" (détaillé par la suite)
user.login=catalogws
user.password=mdp
```

4.3.2 Le fichier droits.xml

Il est possible de nommer ce fichier autrement. Il faut dans ce cas modifier le fichier *onixcatalogws.properties* en conséquences.

Ce fichier doit avoir la structure suivante :

```
<users>

  <profil name="public">
    <pass type="R">
      <path>//Product/RecordReference</path>
      <path>//Product/NotificationType</path>
      <path>//Product/ProductIdentifier[ProductIDType='02']</path>
      <path>//Product/ProductForm</path>
      <path>//Product/Title</path>
      <path>//Product/Website</path>
      <path>//Product/Contributor</path>
      <path>//Product/NumberOfPages</path>
      <path>//Product/Subject</path>
      <path>//Product/OtherText</path>
      <path>//Product/MediaFile</path>
      <path>//Product/Imprint</path>
      <path>//Product/Publisher</path>
      <path>//Product/PublicationDate</path>
      <path>//Product/SupplyDetail/Price</path>
       ...
    </pass>
  </profil>

  <user login="catalogws" password="mdp" lastname="SERVICE"
      firstname="Web"
    function="service web" websiteManagement="false">
    <profilName>public</profilName>
  </user>

</users>
```

Explications Le profil *public* est constitué d'un élément <*pass*> de type "R" signifiant qu'il possède des droits en lecture sur tous les éléments <*path*> qu'il contient.

Il nous est ainsi permis, de décider quels champs ONIX seront délivrés par le web service lors de son interrogation distante.

L'élément <*user*> correspond au nom d'utilisateur du service web (Les *user.login* et *user.password* du fichier *onixcatalogws.properties*)

L'élément <*profilName*> renvoie ici au profil *public*.

4.3.3 Pour finir

Pour finir redémarrer **Tomcat** pour qu'il prenne en compte les modifications.